

Lecture 3: Data Model 설계

김 강 희

khkim@ssu.ac.kr

목 차

● 이론 :

- ❖ 설계 로드맵 구상
- ❖ Source tree 구상
- ❖ 데이터 모델 구상
- ❖ 데이터 모델 코딩

● 실습 :

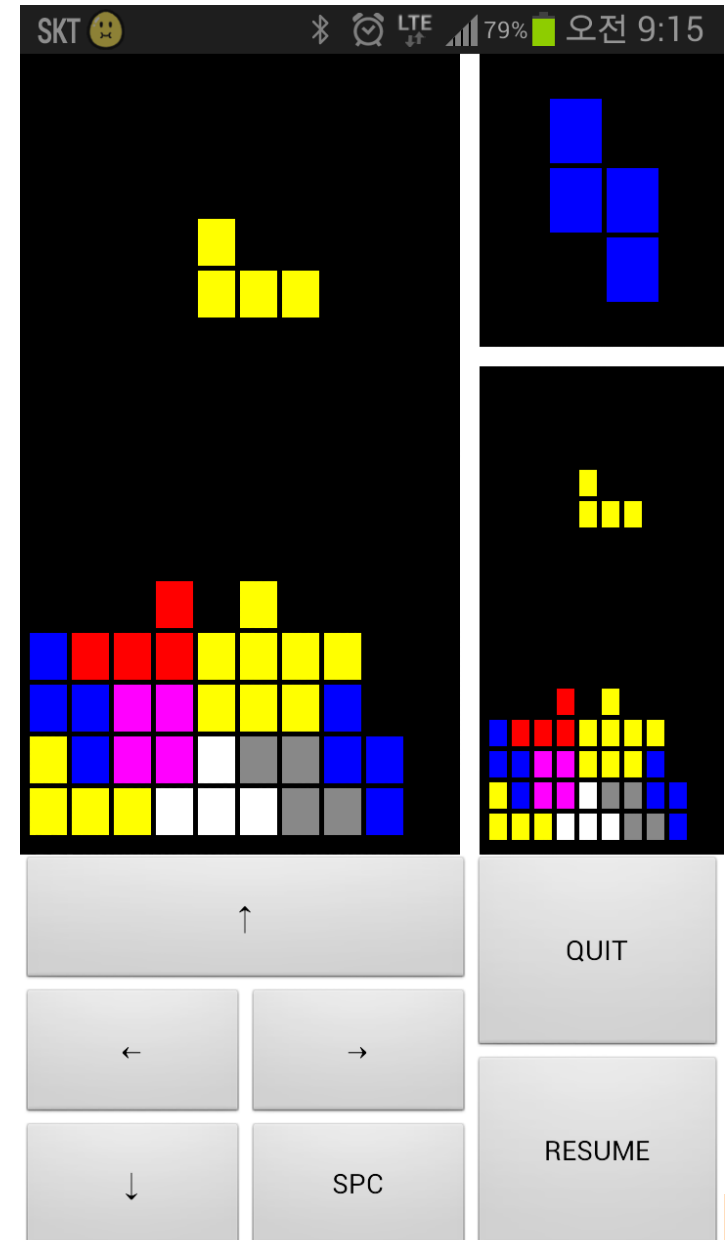
- ❖ v1: 단순 시나리오
- ❖ v2: 확장 시나리오 (다양한 키 입력)
- ❖ v3: 확장 시나리오 (충돌 테스트)
- ❖ v4: 확장 시나리오 (충돌 테스트)
- ❖ v5: 확장 시나리오 (코드 가독성 개선)
- ❖ v6: 확장 시나리오 (충돌 테스트 및 배경 화면 갱신)
- ❖ v7: 확장 시나리오 (충돌 테스트 및 게임 종료)
- ❖ v8 - v11: 과제

이번 Lecture에서 알아야 할 사항들

- 프로젝트 구상 단계의 세부 단계들
 1. 설계 로드맵 구상 : 1인용 흑백 콘솔 테트리스 → 2인용 컬러 GUI 테트리스
 2. Source tree 구상 : deterministic 요소(데이터 모델)과 non-deterministic 요소(외부 인터페이스)로 나누어서 설계
 3. 데이터 모델 구상 : 시나리오들의 집합화(결과물: 순서도)와 시나리오 연산화(결과물: 객체간 연산 정의) 위주로 구상
 4. 데이터 모델 코딩 : 단순 시나리오 → 시나리오 확장
 - 단순 시나리오 (객체 하나 + 시나리오 하나) → 확장 시나리오 코딩 (객체 다수 + 시나리오 다수)
 - 데이터 모델의 단순성을 추구해야 함!!
 - ❖ 경계 조건의 중요성 인식 (mistake-proof 코딩을 위해)

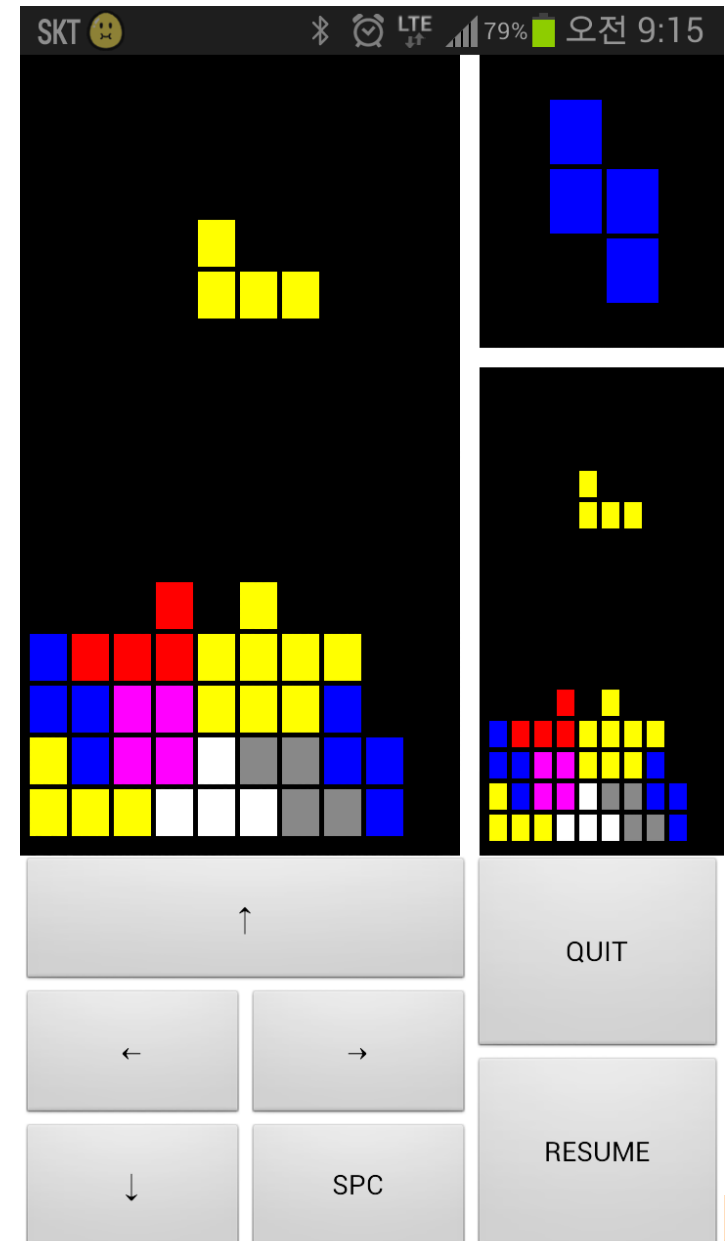
단계 1. 설계 로드맵 구상

- 콘솔 환경
 - ❖ 1인용 흑백 테트리스
 - ❖ 1인용 흑백 테트리스 with a duplicated screen
- 안드로이드 환경
 - ❖ 1인용 컬러 테트리스 with a duplicated screen
 - ❖ 1인용 컬러 테트리스 with Echo server
 - 멀티 쓰레딩 + 소켓
 - ❖ 2인용 컬러 테트리스 with Tetris server
 - Tetris Server 프로그래밍 필요



단계 2. Source tree 구상

- 시스템 측면 (non-deterministic)
 - ❖ 키 입력, 화면 출력, 타이머 구동, 난수 발생 등
- 알고리즘 측면 (deterministic)
 - ❖ 블록 출현 후
 - 이동: 좌, 우, 아래, 추락
 - 회전(90도)
 - ❖ 블록 충돌
 - 좌/우 충돌
 - 아래/추락 충돌
 - 회전 충돌
 - ❖ 행 삭제
 - ❖ 배경 화면 갱신 및 신규 블록 출현

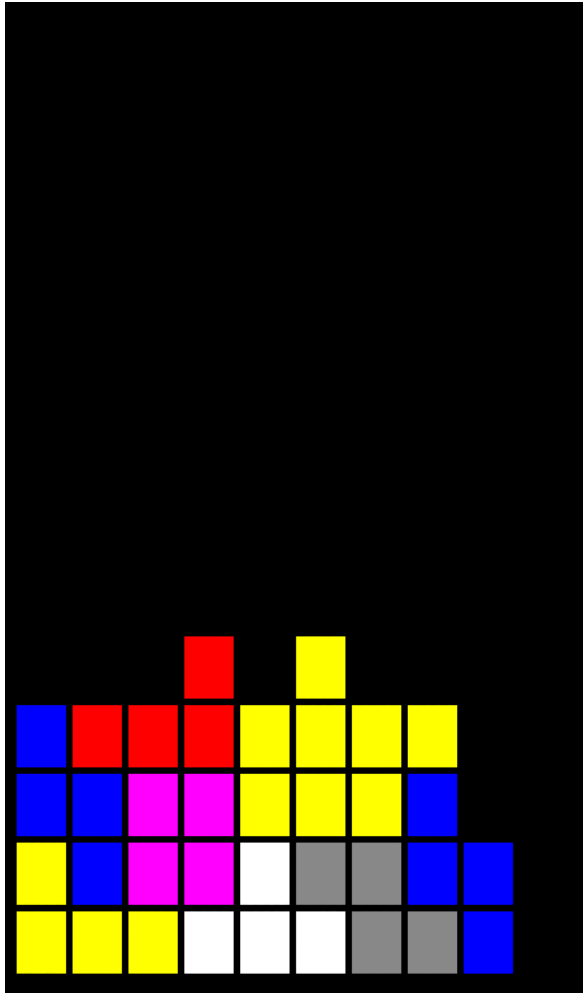


단계 3. 데이터 모델 구상

- 시나리오 연산화 : common scenario 고려 (결과물: 객체 연산의 정의)
 - ❖ 주요 객체들 구상 : 배경, 벽, 쌓인 블록들, 내려오는 블록, ...
 - ❖ 객체 추상화 : 서로 다른 성격의 객체들도 가능한 한 동일한 클래스로 취급할 수 있도록 클래스를 정의 (예: 배경, 벽, 7가지 블록들 → 행렬)
 - 이러한 ‘공격적인’ 추상화가 시나리오 코딩에 미치는 영향을 추후에 신중히 검토해야 함!!
 - ❖ 객체 단순화 : 추상화된 클래스가 너무 복잡한 속성들로 정의되지 않도록 최대한 속성을 단순화 (예: 컬러 블록 → 흑백 블록)
- 시나리오들 집합화 : 모든 시나리오 고려 (결과물: 순서도)
 - ❖ 가능한 시나리오들을 순서도 형태로 열거한다.
 - ❖ 각 시나리오를 선택하여 가급적이면 동일한 타입의 객체의 연산으로 표현한다.

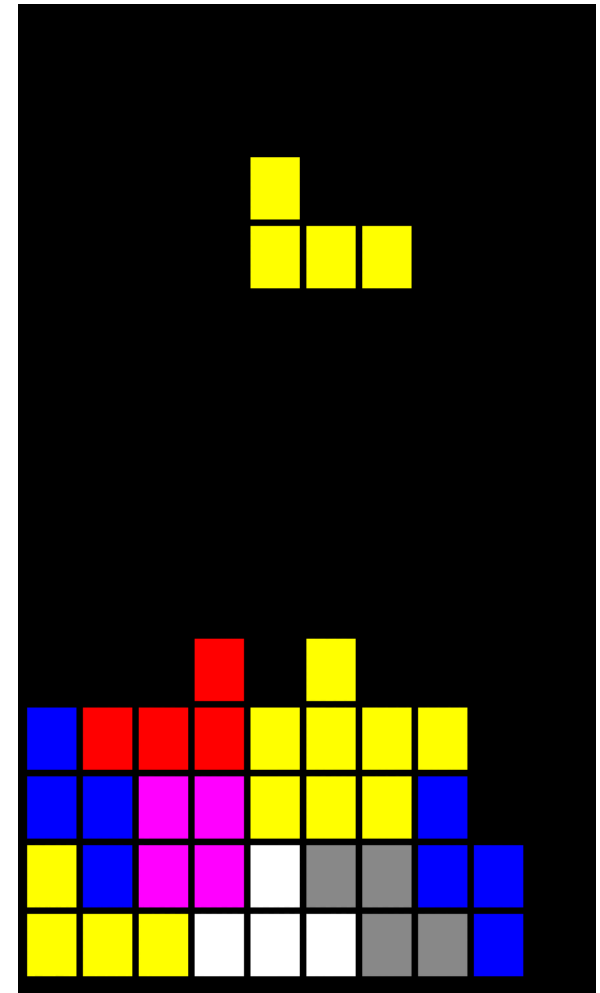
단계 3a : common scenario 연산화

배경



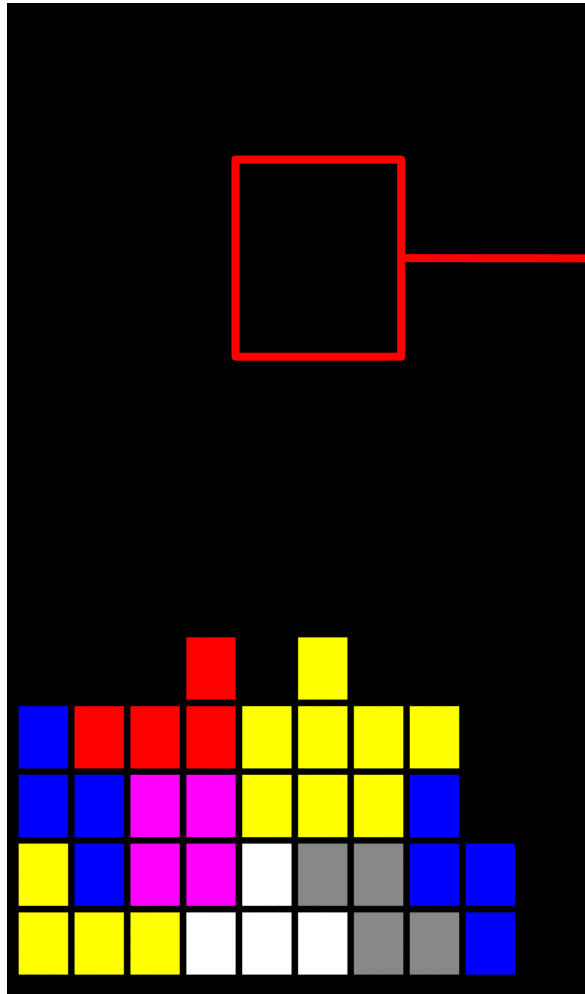
$$+ \begin{array}{c} \text{블록} \\ \begin{array}{|c|c|c|} \hline \text{yellow} & & \\ \hline \text{yellow} & \text{yellow} & \text{yellow} \\ \hline \end{array} \end{array} =$$

출력

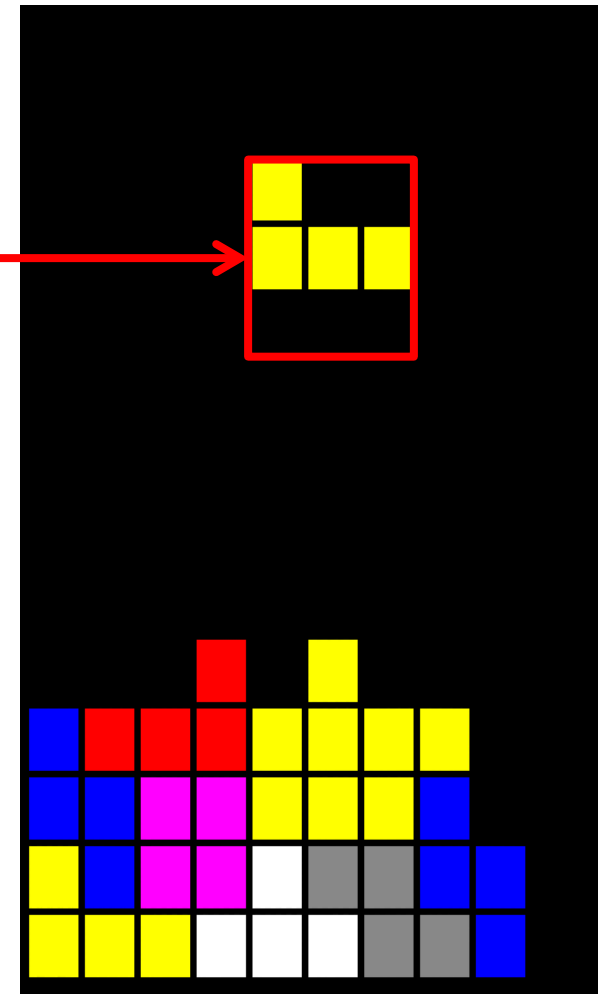


단계 3a : common scenario 연산화

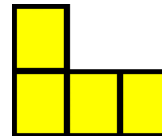
배경



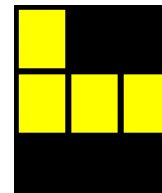
출력



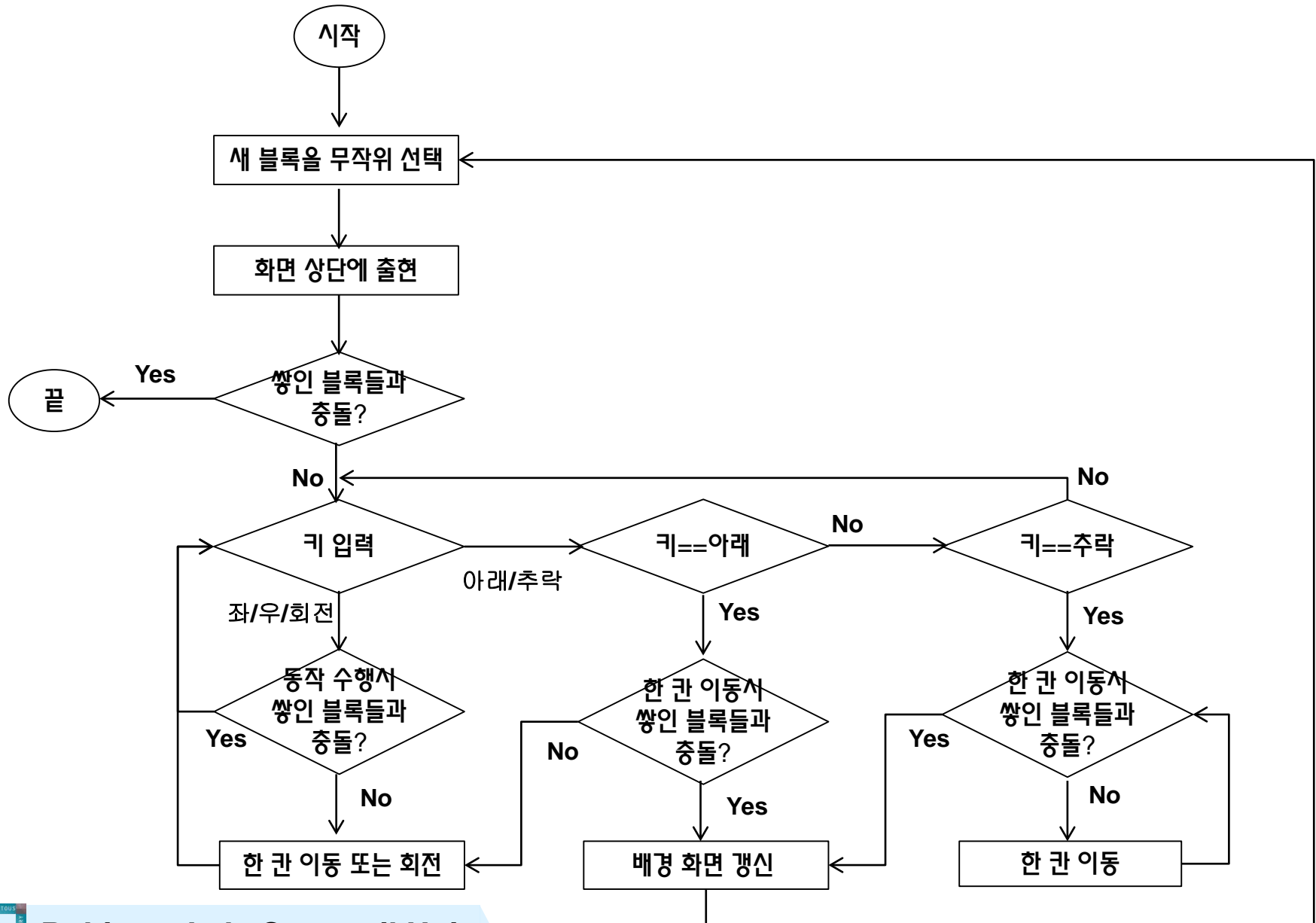
+



=



단계 3b : 모든 시나리오들을 순서도로 표현



단계 4. 데이터 모델 코딩

- 단순 시나리오 코딩 : main 함수 안에서 Matrix class를 이용함
 - ❖ 콘솔 입출력(사용자 인터페이스)은 시스템 측면의 코딩이므로 Matrix.java 파일과 분리된 TestMain.java 파일로 작성함
 - ❖ 난수 발생 또는 키 입력과 같은 non-deterministic 요소는 고정된 값을 갖는다고 가정함
 - ❖ 데이터 모델 설계의 초기 단계부터 콘솔 프린트(printf or println)를 이용하여 모델을 구상, 보정하는 것은 필수적임
- 확장 시나리오 코딩 : Matrix class의 메소드들을 최대한 활용함
 - ❖ 추상화된 Matrix 객체들로 모든 시나리오가 해당 객체들의 연산들로 표현 가능한지를 실제로 코딩하면서 검증함 → 그 결과 만들어진 것이 Matrix 클래스임
 - 단순 시나리오 : 객체 하나 + 시나리오 하나
 - 확장 시나리오 : 객체 다수 + 시나리오 다수
 - ❖ 데이터 모델의 단순성을 추구해야 함!!
 - Matrix 객체를 이용하여 배경, 벽, 쌓인 블록, 떨어지는 블록을 모두 표현할 수 있고, 그들 사이의 연산들을 몇 가지로 정의할 수 있어야 함
 - 객체를 반복 사용함으로 인해서 버려지는 객체들이 발생할 수 있으며 이로 인한 메모리 낭비는 데이터 모델 설계 단계에서 코딩의 생산

단계 4a. 단순 시나리오 코딩

```
int top = 0; // 상태 변수
int left = 4; // 상태 변수
Matrix iScreen = new Matrix(20,10); // 상태 변수
Matrix currBlk = new Matrix(arrayBlock); // 상태 변수

Matrix tempBlk = iScreen.clip(top, left,
                             top+currBlk.dy, left+currBlk.dx); // 임시 변수
tempBlk = tempBlk.add(currBlk);

Matrix oScreen = new Matrix(iScreen); // 상태 변수
oScreen.paste(tempBlk, top, left);
```

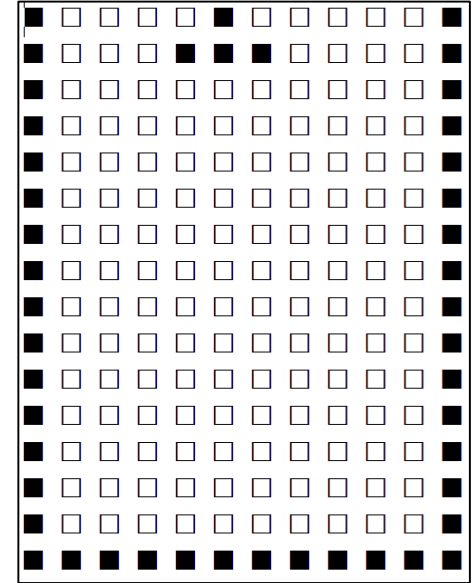
Main.java (v1)

참고: printMatrix 메소드는 drawMatrix와 동일한 메소드임!!

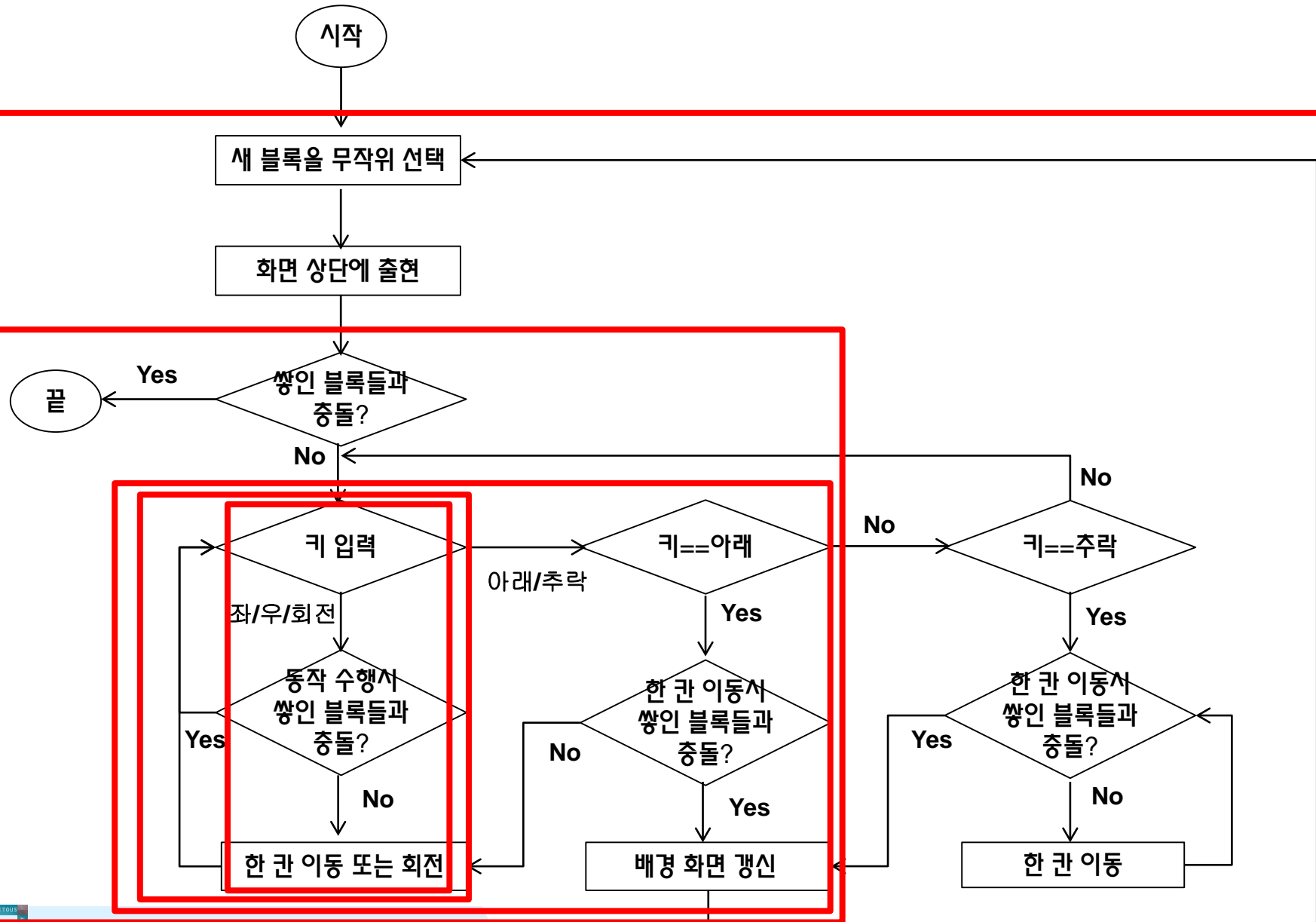
```

2 public class Main {
3+ public static void printMatrix(Matrix blk) {
16- public static void main(String[] args) throws Exception {
17     int[][] arrayScreen = {
18         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
19         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
20         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
21         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
22         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
23         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
24         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
25         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
26         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
27         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
28         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
29         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
30         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
31         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
32         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
33         { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
34     };
35     int[][] arrayBlk = {
36         { 0, 1, 0 },
37         { 1, 1, 1 },
38         { 0, 0, 0 },
39     };
40     int top = 0, left = 4;
41     Matrix iScreen = new Matrix(arrayScreen);
42     Matrix currBlk = new Matrix(arrayBlk);
43     Matrix tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
44     tempBlk = tempBlk.add(currBlk);
45     Matrix oScreen = new Matrix(iScreen);
46     oScreen.paste(tempBlk, top, left);
47     printMatrix(oScreen); System.out.println();
48 }

```



단계 4b. 확장 시나리오 코딩 (안쪽 → 바깥쪽)



단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
 - ❖ 키입력 기능 구현을 위해서 getKey method 추가
 - 이로 인해 파일 첫머리에는 import 구문들 추가
 - getKey 메소드 작성을 위한 정보는 인터넷에서 쉽게 찾을 수 있음
 - ❖ 코드 가독성을 위해서 arrayScreen, arrayBlock 변수를 main 밖으로 이동
 - main 메소드는 static이므로 static 키워드를 변수 앞에 추가

Main.java (v2)

```

1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4
5 public class Main {
6     static int[][] arrayScreen = {
24 static int[][] arrayBlk = {
29 public static void printMatrix(Matrix blk) {
42 private static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
43 private static String line = null;
44 private static int nKeys = 0;
45 private static char getKey() throws IOException {
46     char ch;
47     if (nKeys != 0) { // 이전 getKey() 호출의 결과 line 배열에 남아 있는 원소가 있는지 확인한다.
48         ch = line.charAt(line.length() - nKeys);
49         nKeys--; // 남아있는 원소들 중에서 첫번째 원소를 ch에 저장하고 리턴한다.
50         return ch;
51     }
52     do {
53         line = br.readLine(); // 자바 콘솔에서 한 라인을 읽는다.
54         nKeys = line.length();
55     } while (nKeys == 0); // 적어도 하나의 key가 입력될 때까지 반복한다.
56     ch = line.charAt(0); // line 배열의 첫번째 원소를 ch에 저장하고 리턴한다.
57     nKeys--;
58     return ch;
59 }

```

// getKey() 메소드에서 요구하는 패키지들

// main() 안에서 정의된 것을 밖으로 옮기면서 static 키워드 추가함

// 키 입력을 얻어오는 메소드

Main.java (v2)

```

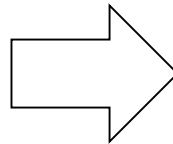
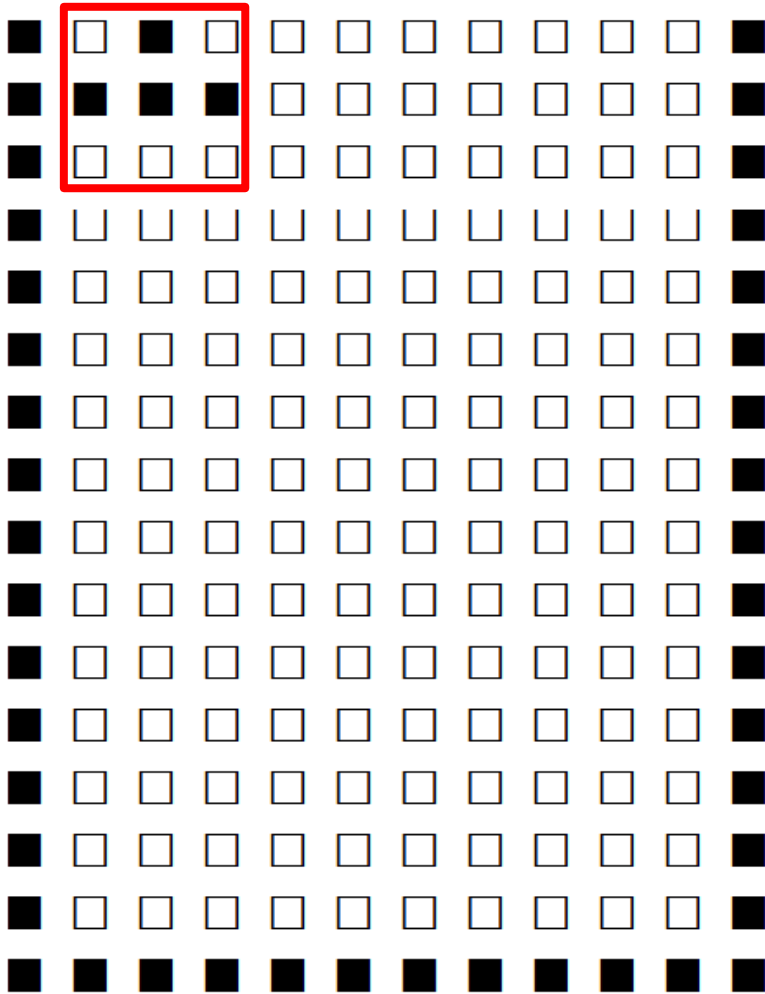
60 public static void main(String[] args) throws Exception {
61     int top = 0, left = 4;
62     char key;
63     Matrix iScreen = new Matrix(arrayScreen);
64     Matrix currBlk = new Matrix(arrayBlk);
65     Matrix tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
66     tempBlk = tempBlk.add(currBlk);
67     Matrix oScreen = new Matrix(iScreen);
68     oScreen.paste(tempBlk, top, left);
69     printMatrix(oScreen); System.out.println();
70
71     while ((key = getKey()) != 'q') {
72         switch(key) {
73             case 'a': left--; break; // move left
74             case 'd': left++; break; // move right
75             case 's': top++; break; // move down
76             case 'w': break; // rotate the block
77             case ' ': break; // drop the block
78             default: System.out.println("unknown key!");
79         }
80         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
81         tempBlk = tempBlk.add(currBlk);
82         oScreen = new Matrix(iScreen);
83         oScreen.paste(tempBlk, top, left);
84         printMatrix(oScreen); System.out.println();
85     }
86 }
87 }

```

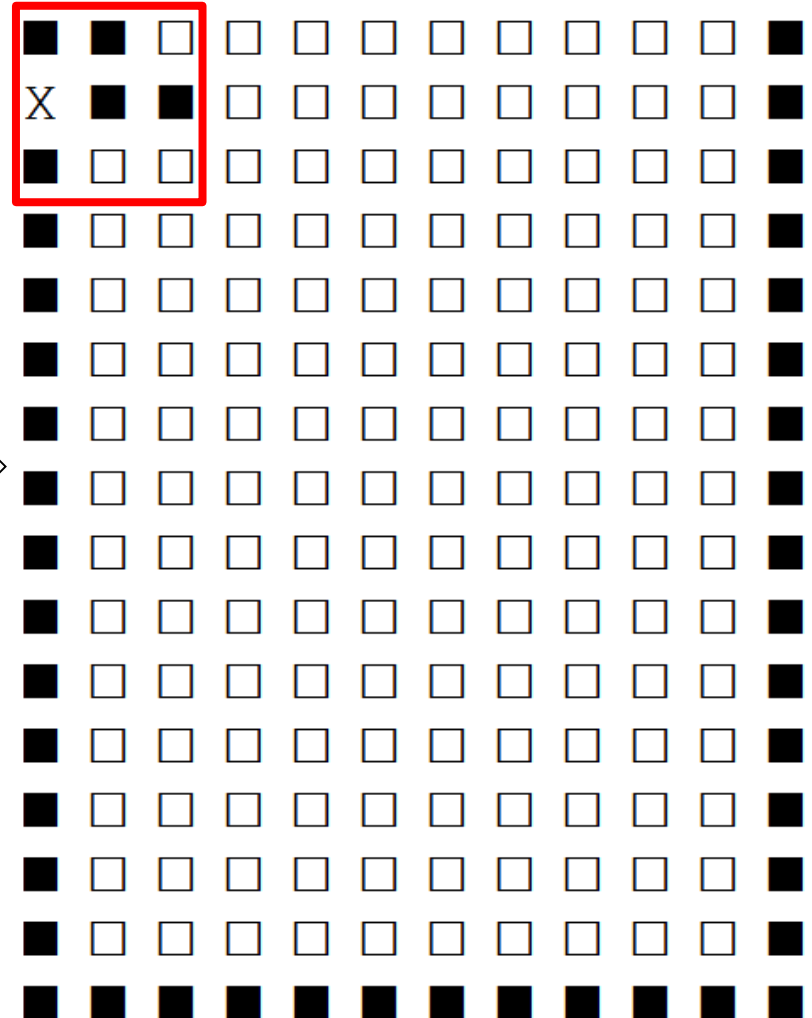
// 반복되는 코드 (단, tempBlk, oScreen 객체를 다시 선언하지 않음)

Main.java (v2) : 벽에 충돌하면?

a



a



단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
- Left/Right/Down키 입력시 충돌 있으면 블록 제자리 위치
 - ❖ 블록 경계에서 충돌이 발생하면? anyGreaterThan 메소드 + 배경 화면 배열에 벽을 한 겹으로 표현 + 변경된 (top, left) 좌표를 복원함 → v3

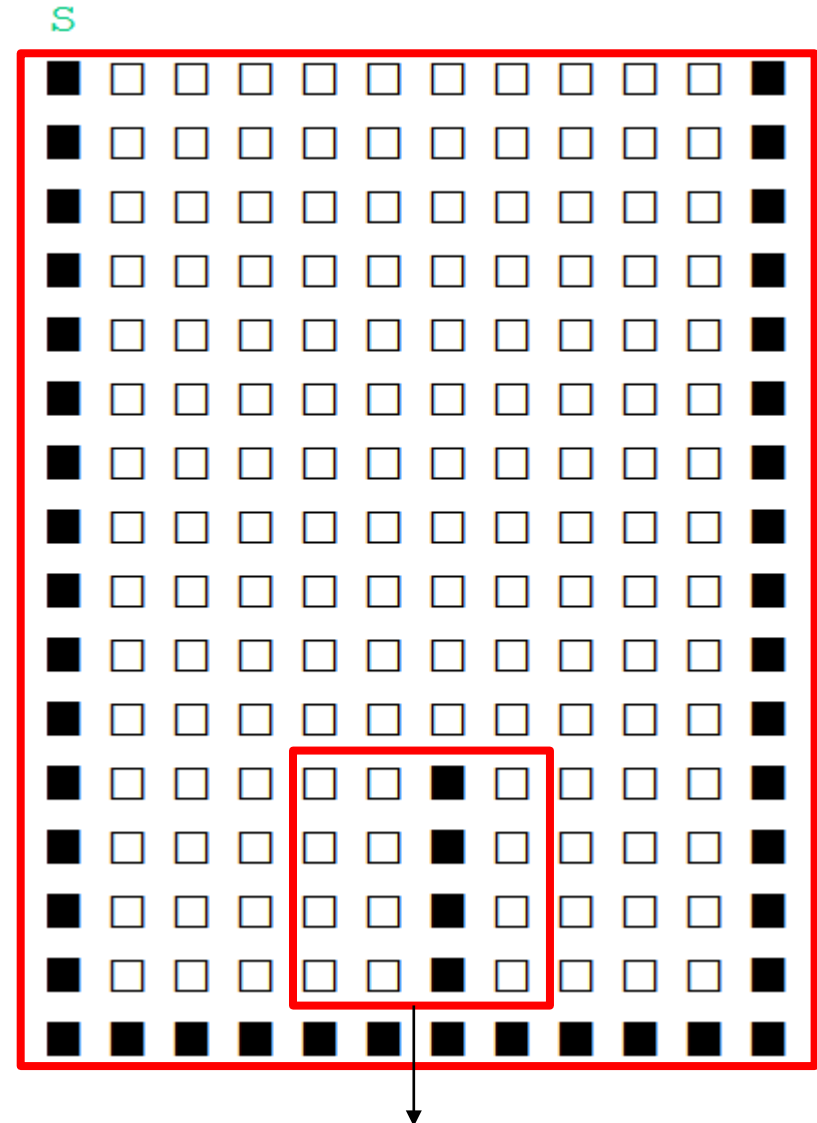
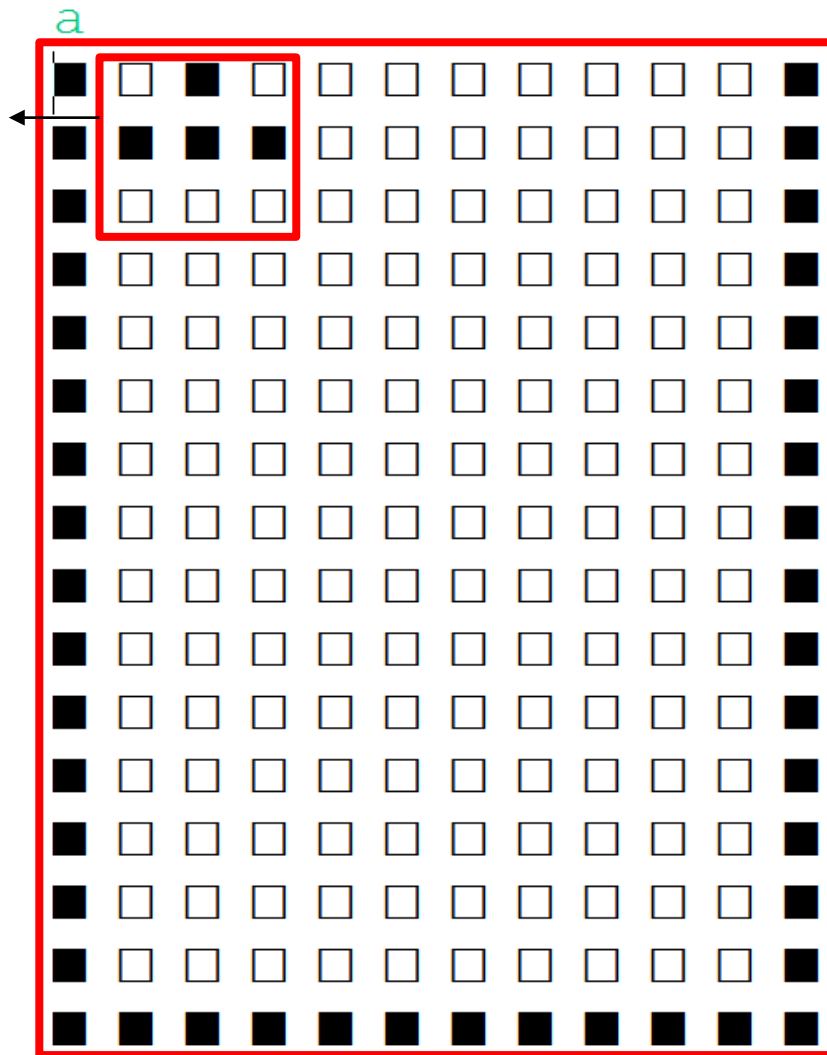
Main.java (v3)

```

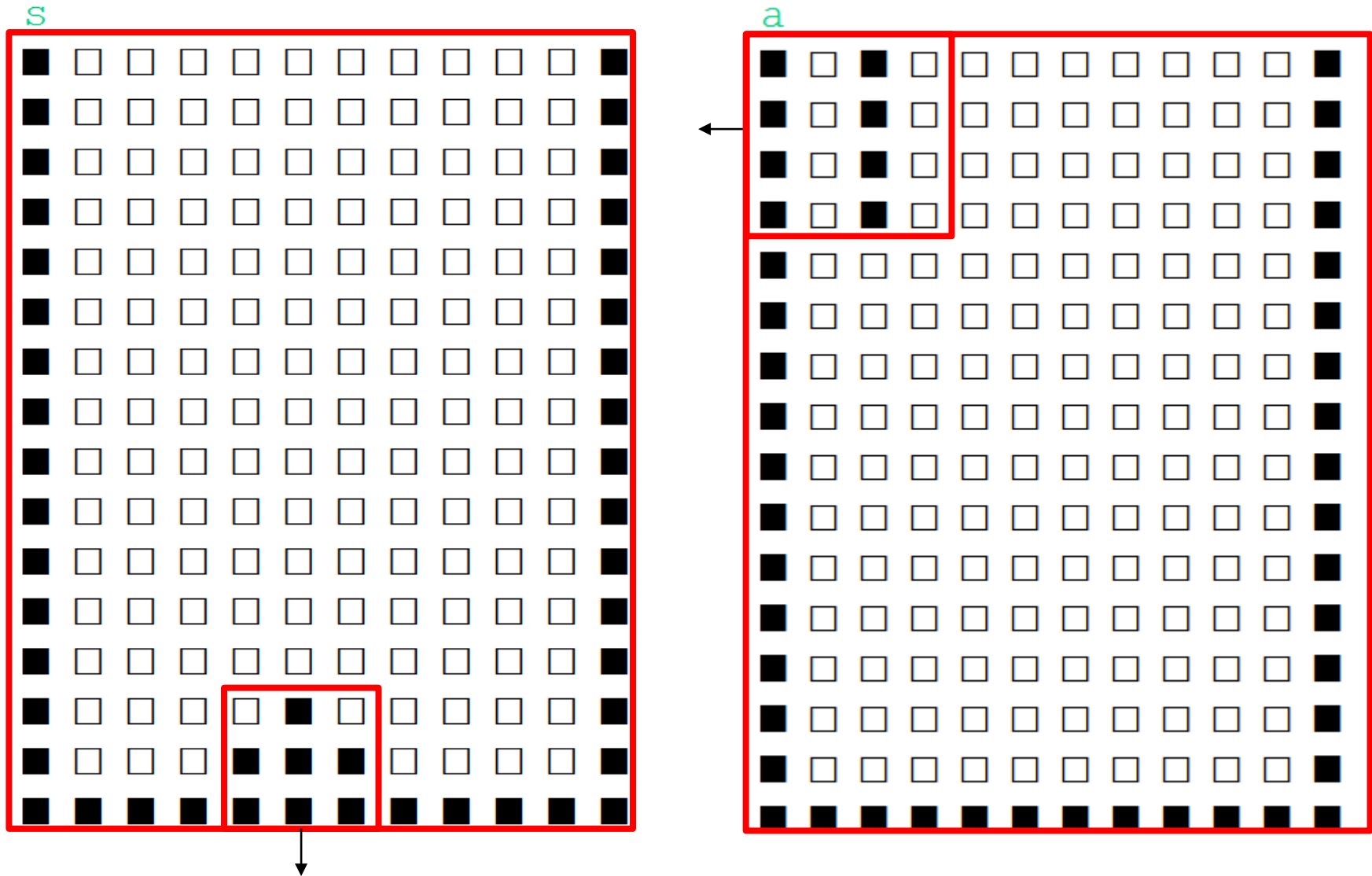
60 public static void main(String[] args) throws Exception {
61     int top = 0, left = 4;
62     char key;
63     Matrix iScreen = new Matrix(arrayScreen);
64     Matrix currBlk = new Matrix(arrayBlk);
65     Matrix tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
66     tempBlk = tempBlk.add(currBlk);
67     Matrix oScreen = new Matrix(iScreen);
68     oScreen.paste(tempBlk, top, left);
69     printMatrix(oScreen); System.out.println();
70     while ((key = getKey()) != 'q') {
71         switch(key) {
72             case 'a': left--; break; // move left
73             case 'd': left++; break; // move right
74             case 's': top++; break; // move down
75             case 'w': break; // rotate the block clockwise
76             case ' ': break; // drop the block
77             default: System.out.println("unknown key!");
78         }
79         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
80         tempBlk = tempBlk.add(currBlk);
81         if (tempBlk.anyGreaterThan(1)) { // 벽 충돌시 undo 수행
82             switch(key) {
83                 case 'a': left++; break; // undo: move right
84                 case 'd': left--; break; // undo: move left
85                 case 's': top--; break; // undo: move up
86                 case 'w': break; // undo: rotate the block counter-clockwise
87                 case ' ': break; // undo: move up
88             }
89             tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
90             tempBlk = tempBlk.add(currBlk);
91         }
92         oScreen = new Matrix(iScreen);
93         oScreen.paste(tempBlk, top, left);
94         printMatrix(oScreen); System.out.println();
95     }
96 }

```

Main.java (v3) : 블록 경계에서 충돌이 발생하는 경우?



Main.java (v3) : 블록 내부에서 충돌이 발생하는 경우?



```
Exception in thread "main" MatrixException: invalid matrix range
    at Matrix.clip(Matrix.java:53)
    at Main.main(Main.java:80)
```

단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
- Left/Right/Down키 입력시 충돌 있으면 블록 제자리 위치
 - ❖ 블록 경계에서 충돌이 발생하면? anyGreaterThan 메소드 + 배경 화면 배열에 벽을 한 겹으로 표현 + 변경된 (top, left) 좌표를 복원함 → v3
 - ❖ 블록 내부에서 충돌이 발생하면 ? anyGreaterThan 메소드 + 배경 화면 배열에 벽을 여러 겹으로 표현 + 변경된 (top, left) 좌표를 복원함 → v4

Main.java (v4)

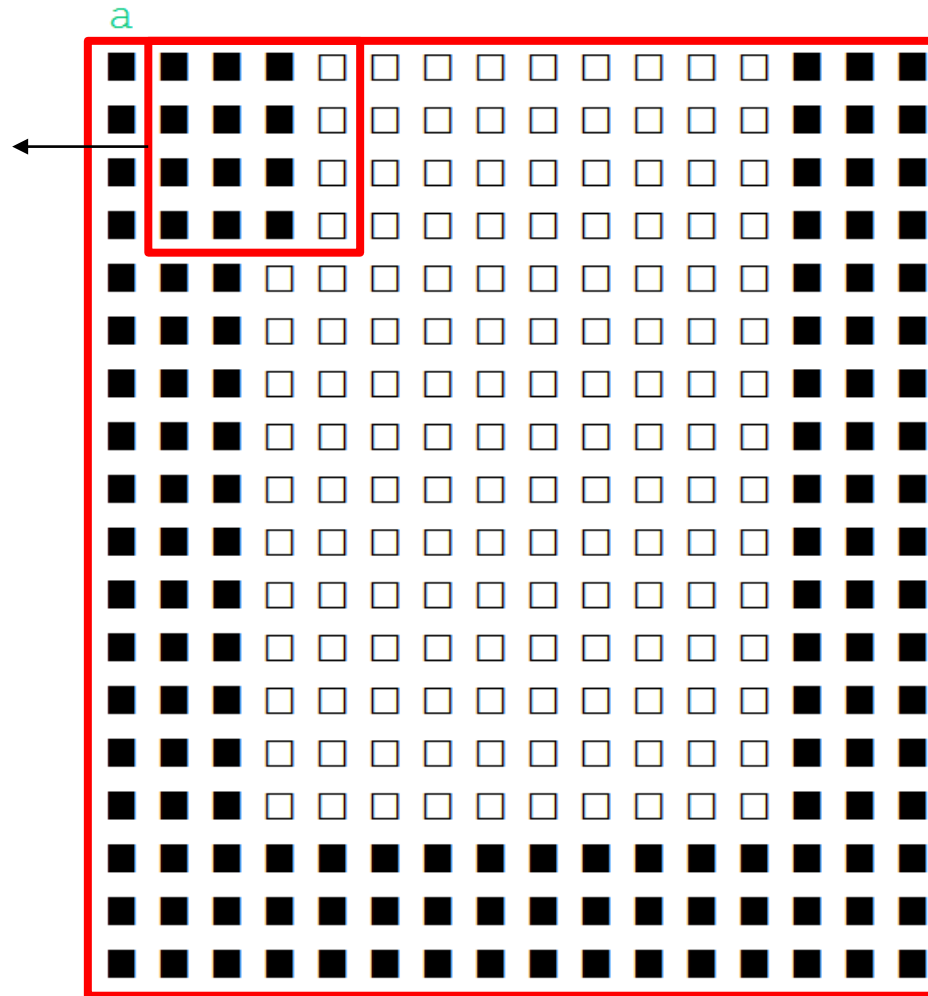
- arrayScreen 배열에서 벽을 여러 겹으로 정의함 → 몇 겹으로?

```

5 public class Main {
6     static int[][] arrayScreen = { // array[15+3][10+6]
7         { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
8         { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
9         { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
10        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
11        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
12        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
13        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
14        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
15        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
16        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
17        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
18        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
19        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
20        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
21        { 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 },
22        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
23        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
24        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
25    };

```

Main.java (v4)



단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
- Left/Right/Down키 입력시 충돌 있으면 블록 제자리 위치
 - ❖ 블록 경계에서 충돌이 발생하면? anyGreaterThan 메소드 + 배경 화면 배열에 벽을 한 겹으로 표현 + 변경된 (top, left) 좌표를 복원함 → v3
 - ❖ 블록 내부에서 충돌이 발생하면 ? anyGreaterThan 메소드 + 배경 화면 배열에 벽을 여러 겹으로 표현 + 변경된 (top, left) 좌표를 복원함 → v4
 - 여러 겹의 벽을 가진 배경화면 배열의 중요성을 강조하기 위해 이러한 배열을 생성하는 createArrayScreen 메소드를 정의함 → v5

Main.java (v5)

```

5 public class Main {
6+ static int[][] arrayBlk = {};
12 private static int iScreenDy = 15;
13 private static int iScreenDx = 10;
14 private static int iScreenDw = 4; // large enough to cover the largest block
15- private static int[][] createArrayScreen(int dy, int dx, int dw) {
16     int y, x;
17     int[][] array = new int[dy + dw][dx + 2*dw];
18     for (y = 0; y < array.length; y++)
19         for (x = 0; x < dw; x++)
20             array[y][x] = 1;
21     for (y = 0; y < array.length; y++)
22         for (x = dw + dx; x < array[0].length; x++)
23             array[y][x] = 1;
24     for (y = dy; y < array.length; y++)
25         for (x = 0; x < array[0].length; x++)
26             array[y][x] = 1;
27     return array;
28 }
29- public static void printScreen(Matrix screen) {
30     int dy = screen.get_dy();
31     int dx = screen.get_dx();
32     int dw = iScreenDw;
33     int array[][] = screen.get_array();
34     for (int y = 0; y < dy - dw + 1; y++) {
35         for (int x = dw - 1; x < dx - dw + 1; x++) {
36             if (array[y][x] == 0) System.out.print("□ ");
37             else if (array[y][x] == 1) System.out.print("■ ");
38             else System.out.print("X ");
39         }
40         System.out.println();
41     }
42 }

```

Main.java (v5)

```

74 public static void main(String[] args) throws Exception {
75     int top = 0;
76     int left = iScreenDw + iScreenDx/2 - 2;
77     int[][] arrayScreen = createArrayScreen(iScreenDy, iScreenDx, iScreenDw);
78     char key;
79     Matrix iScreen = new Matrix(arrayScreen);
80     Matrix currBlk = new Matrix(arrayBlk);
81     Matrix tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
82     tempBlk = tempBlk.add(currBlk);
83     Matrix oScreen = new Matrix(iScreen);
84     oScreen.paste(tempBlk, top, left);
85     printScreen(oScreen); System.out.println();
86     while ((key = getKey()) != 'q') {
87         switch(key) {
88             case 'a': left--; break; // move left
89             case 'd': left++; break; // move right
90             case 's': top++; break; // move down
91             case 'w': break; // rotate the block clockwise
92             case ' ': break; // drop the block
93             default: System.out.println("unknown key!");
94         }
95         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
96         tempBlk = tempBlk.add(currBlk);
97         if (tempBlk.anyGreaterThan(1)) {
98             switch(key) {
99                 case 'a': left++; break; // undo: move right
100                case 'd': left--; break; // undo: move left
101                case 's': top--; break; // undo: move up
102                case 'w': break; // undo: rotate the block counter-clockwise
103                case ' ': break; // undo: move up
104            }
105            tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
106            tempBlk = tempBlk.add(currBlk);
107        }
108        oScreen = new Matrix(iScreen);
109        oScreen.paste(tempBlk, top, left);
110        printScreen(oScreen); System.out.println();
111    }
112 }

```

단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
- Left/Right/Down키 입력시 충돌 있으면 블록 제자리 위치 → v5
- Down키 입력시 충돌 있으면 배경 화면 갱신 → v6

```

74 public static void main(String[] args) throws Exception {
75     boolean newBlockNeeded = false;

76     while ((key = getKey()) != 'q') {
77         switch(key) {
78             case 'a': left--; break; // move left
79             case 'd': left++; break; // move right
80             case 's': top++; break; // move down
81             case 'w': break; // rotate the block clockwise
82             case ' ': break; // drop the block
83             default: System.out.println("unknown key!");
84         }
85         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
86         tempBlk = tempBlk.add(currBlk);
87         if (tempBlk.anyGreaterThan(1)) {
88             switch(key) {
89                 case 'a': left++; break; // undo: move right
90                 case 'd': left--; break; // undo: move left
91                 case 's': top--; newBlockNeeded = true; break; // undo: move up
92                 case 'w': break; // undo: rotate the block counter-clockwise
93                 case ' ': break; // undo: move up
94             }
95             tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
96             tempBlk = tempBlk.add(currBlk);
97         }
98         oScreen = new Matrix(iScreen);
99         oScreen.paste(tempBlk, top, left);
100         printScreen(oScreen); System.out.println();
101         if (newBlockNeeded) {
102             iScreen = new Matrix(oScreen);
103             top = 0; left = iScreenDw + iScreenDx/2 - 2;
104             newBlockNeeded = false;
105             currBlk = new Matrix(arrayBlk);
106             tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
107             tempBlk = tempBlk.add(currBlk);
108             oScreen = new Matrix(iScreen);
109             oScreen.paste(tempBlk, top, left);
110             printScreen(oScreen); System.out.println();
111         }
112     }
113 }

```

// 바닥 충돌시 새 블록 출현

단계 4b. 확장 시나리오 코딩

- Left/Right/Down키 입력시 충돌 없으면 블록 한 칸 이동 → v2
- Left/Right/Down키 입력시 충돌 있으면 블록 제자리 위치 → v5
- Down키 입력시 충돌 있으면 배경 화면 갱신 → v6
- 새 블록이 출현하자마다 충돌 있으면 게임 종료 → v7

Main.java (v7)

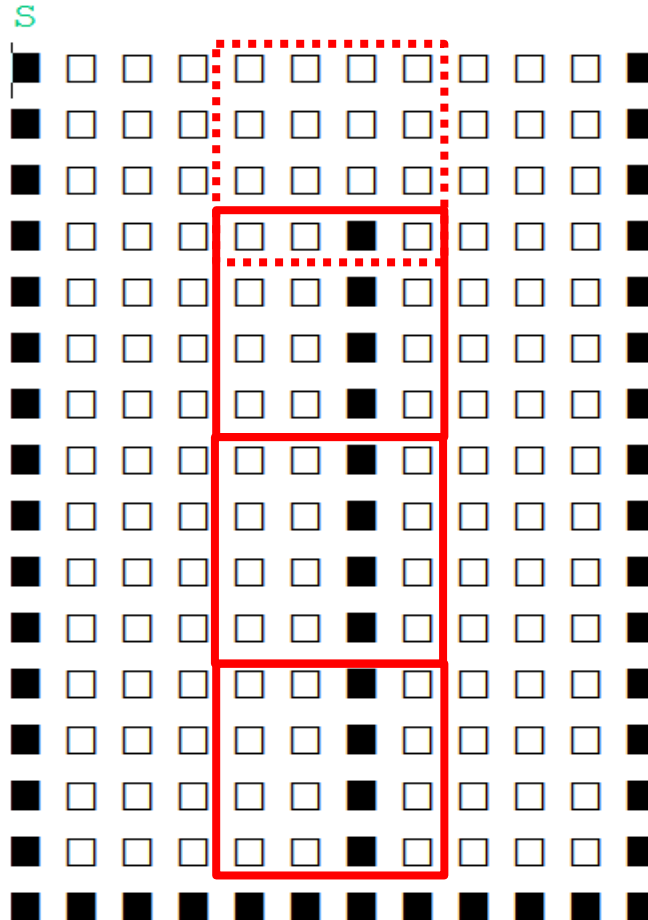
```

109     oScreen = new Matrix(iScreen);
110     oScreen.paste(tempBlk, top, left);
111     printScreen(oScreen); System.out.println();
112     if (newBlockNeeded) {
113         iScreen = new Matrix(oScreen);
114         top = 0; left = iScreenDw + iScreenDx/2 - 2;
115         newBlockNeeded = false;
116         currBlk = new Matrix(arrayBlk);
117         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx());
118         tempBlk = tempBlk.add(currBlk);
119         if (tempBlk.anyGreaterThan(1)) {
120             System.out.println("Game Over!");
121             System.exit(0);
122         }
123         oScreen = new Matrix(iScreen);
124         oScreen.paste(tempBlk, top, left);
125         printScreen(oScreen); System.out.println();
126     }
127 }
128 }
129 }

```

// 새 블록 출현시 충돌 있으면
게임 종료

Main.java (v7)



Game Over!

단계 4b. 확장 시나리오 코딩 : 남은 숙제들

- 7가지 블록의 무작위 선택 → v8
 - ❖ 7가지 블록들을 Matrix 객체들로 미리 생성함
 - ❖ 이 객체들을 필요할 때마다 난수를 발생시켜 선택함

```
import java.util.Random
Random random = new Random(); // 프로그램 시작시 한번만 호출
int idxBlockType = random.nextInt(7); // idxBlockType은 상태 변수
Matrix currBlk = setOfBlockObjects[idxBlockType];
```
- Rotate키 처리: Rotate의 결과를 표현하는 Matrix 객체들을 미리 생성함 → v9


```
idxBlockDegree = (idxBlockDegree + 1) % 4; // idxBlockDegree은 상태 변수
Matrix currBlk = setOfBlockObjects[idxBlockType][idxBlockDegree];
```
- Space키 처리 (블록 추락) → v10
 - ❖ 바닥에 충돌할 때까지 아래로 한 칸씩 이동함을 루프 형태로 반복함
 - ❖ 한 칸씩 이동한 결과 생성되는 oScreen을 화면에 출력하지 말되, 마지막으로 블록이 바닥에 인접한 장면의 oScreen은 출력해야 함
- Full line 삭제 → v11
 - ❖ Down키 입력시 충돌 있고, full line들이 발견되면 해당 line들을 배경 화면에서 삭제함 : Matrix class의 sum, clip, paste 메소드 이용하고, screen 배열의 검색 범위를 줄이는 것에 유의해야 함